

# Capstone Project

## My Aadhar Application

### Sprints

#### Sprint1

Creating the basic setups for the backend .And performing the backend operations with spring boot with MySQL create the required tables.

#### Sprint2

Creating the basic setups for the frontend using angular,connecting the backend to the front end to display the outcome of each operation

#### Sprint3

Writing a testng script to perform valid test cases and automate it using jenkins

#### Sprint4

Deploying spring boot project.jar file in AWS instance, and dockerising the image even automating the dockerisation using jenkins and upload the docker image to docker hub.

### **FrontEnd**

#### **Service.ts**

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { UserDetails } from 'src/classes/details';
import { Logincred } from 'src/classes/logincredentials';
import { RegisterUser } from 'src/classes/registercred';

@Injectable({
  providedIn: 'root'
})
export class AllserviceService {

  private backednURL="http://localhost:8080/citizen";
  constructor(private http:HttpClient) { }

  Logincred():Observable<Logincred[]> {
```

```

        return this.http.get<Logincred[]>(`${this.backednURL}/userLogin`);
    }

    viewDetails():Observable<UserDetails[]>{
        return
this.http.get<UserDetails[]>(`${this.backednURL}/viewDetails`);
    }

    adduser(name:string,dob:string,address:string,emailid:string,phno:numero
r,gender:string,):Observable<UserDetails> {
        const
newuser:UserDetails={name:name,dob:dob,address:address,emailid:emailid,
phno:phno,gender:gender};
        return
this.http.post<UserDetails>(`${this.backednURL}/addCitizen`,newuser)
    }
    registeruser(emailid:string,pwd:string):Observable<RegisterUser>{
        const register:RegisterUser={emailid:emailid,pwd:pwd};
        return
this.http.post<RegisterUser>(`${this.backednURL}/userRegister`,register
);
    }
    registeredusers():Observable<RegisterUser[]>{
        return
this.http.get<RegisterUser[]>(`${this.backednURL}/getUsers`);
    }
    removeuser(phno:number):Observable<UserDetails>{
        return
this.http.delete<UserDetails>(`${this.backednURL}/Dismiss/${phno}`);
    }
}

```

## UserDetails

```

export class UserDetails {
    name:string ;
    dob:string;
    address:string;
    emailid:string;
    phno:number;
    gender:string;
    constructor(name:string
,dob:string,address:string,emailid:string,phno:number,gender:string) {

```

```
this.name=name;this.dob=dob;this.address=address;this.emailid=emailid;t  
his.phno=phno;this.gender=gender;  
    }  
}
```

## **BackEnd**

### **Controller.java**

```
package com.sec.controller;
```

```
import java.util.List;
```

```
import java.util.Optional;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.http.MediaType;
```

```
import org.springframework.web.bind.annotation.CrossOrigin;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.PathVariable;
```

```
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
import com.sec.entity.*;
```

```
import com.sec.service.*;
```

```
import com.sec.repo.*;
```

```
@RestController
```

```
@RequestMapping("/citizen")
```

```
@CrossOrigin()
```

```
public class MainController {
```

```
    @Autowired
```

```
    Adminservice ads;
```

```
    @Autowired
```

```
    Userservice urs;
```

```
    Userrepo up;
```

```

    @GetMapping(value="/userLogin", consumes =
MediaType.APPLICATION_JSON_VALUE)
    public String userLogin(@RequestBody Usercredentials u){
        return urs.login(u.getEmailid(), u.getPwd());}

```

```

    @GetMapping(value="/getUsers",produces=
MediaType.APPLICATION_JSON_VALUE)
    public List<Usercredentials> viewusers(){
        return urs.viewusers();}

```

```

    @PostMapping(value="/userRegister", consumes =
MediaType.APPLICATION_JSON_VALUE)
    public String userRegiter(@RequestBody Usercredentials u){
        return urs.userRegister(u);}

```

```

    @PostMapping(value="/addCitizen" , consumes =
MediaType.APPLICATION_JSON_VALUE)
    public String addCitizen(@RequestBody UserEntity u){
        return ads.addcitizen(u);}

```

```

    @GetMapping(value="/Dismiss/{phno}",produces=MediaType.APPLICATION_JSO
N_VALUE)
    public String dismisscitizen(@PathVariable("phno") long phno) {
        return ads.removecitizen(phno);
    }

```

```

    @GetMapping(value="/viewDetails", produces=
MediaType.APPLICATION_JSON_VALUE)
    public List<UserEntity> viewAll(){
        return ads.viewall();}
}

```

### **Testclasses.java** **package com.test;**

```

import org.testng.annotations.Test;
import org.testng.AssertJUnit;
import org.testng.annotations.Test;

```

```

import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;
import org.testng.annotations.BeforeSuite;
import org.testng.Assert;
import org.testng.annotations.AfterSuite;

public class Aadhartest {

    public WebDriver driver;
    public String testURL="http://localhost:8080/citizen/";
    @Test
    public void logintest() {
        driver.get(testURL + "login");
        WebElement uname=driver.findElement(By.id("phoneNumber"));
        uname.sendKeys("admin1");
        WebElement pwd=driver.findElement(By.id("password"));
        pwd.sendKeys("admin01");
        WebElement
btn=driver.findElement(By.xpath("/html/body/app-root/app-login/div/div[1]/div/div/div/form/button"));
        btn.click();
    }

    @Test
    public void admindashboardtest() {
        driver.get(testURL+"admin");
        WebElement
addusnbtn=driver.findElement(By.xpath("/html/body/app-root/app-admin/div/a"));
        addusnbtn.click();
        WebElement name=driver.findElement(By.id("name"));
        WebElement dob=driver.findElement(By.id("dob"));
        WebElement address=driver.findElement(By.id("address"));
        WebElement emailid=driver.findElement(By.id("emailid"));
        WebElement phno=driver.findElement(By.id("phno"));
        WebElement gender=driver.findElement(By.id("gender"));
        WebElement
addbtn=driver.findElement(By.xpath("/html/body/app-root/app-adduser/div/form/button"));
    }
}

```

```

name.sendKeys("AB");dob.sendKeys("23");address.sendKeys("KA");emailid.sendKe
ys("cv@gm.com");phno.sendKeys("8520");gender.sendKeys("M");
    addbtn.click();
    Alert alert=driver.switchTo().alert();
    AssertJUnit.assertEquals(alert.getText(), "AB has been Added" );
    alert.accept();
}

```

@Test

```

public void scrolltest() {
    driver.get(testURL+"admin");
    Actions scrollpage=new Actions(driver);

```

```

scrollpage.sendKeys(Keys.PAGE_DOWN).sendKeys(Keys.PAGE_DOWN).pause(20
00).sendKeys(Keys.PAGE_UP).build().perform();
}

```

@Test

```

public void userpagetest() {
    driver.get(testURL+"userpage");
    WebElement btn1=driver.findElement(By.xpath(""));
    btn1.click();
    Alert alert=driver.switchTo().alert();
    AssertJUnit.assertEquals(alert.getText(), "Request Sent! You will hear from us
soon" );
    alert.accept();
    WebElement btn2=driver.findElement(By.xpath(""));
    btn2.click();
    Alert alert2=driver.switchTo().alert();
    AssertJUnit.assertEquals(alert2.getText(), "Your Update request has been
sent" );
    alert.accept();
}

```

@BeforeSuite

```

public void beforeSuite() {

```

```

System.setProperty("webdriver.chrome.driver","C:\\Users\\Swaroop\\Desktop\\Eclipse
Workspace\\Application-test-aadhar\\chromedriver.exe");
    driver= new ChromeDriver();

```

```
}
```

```
@AfterSuite
```

```
public void afterSuite() {driver.close();}
```

```
}
```

### **Application.properties**

### **Jenkins Pipeline**

```
pipeline{
```

```
    agent any
```

```
    tools {
```

```
        // Install the Maven version configured as "M3" and add it to the path.
```

```
        maven "default"
```

```
        //jdk "java1"
```

```
    }
```

```
    stages {
```

```
        stage('Maven Build') {
```

```
            steps {
```

```
                dir('C:/Users/Swaroop/Desktop/EclipseWorkspace/Backend-aadhar') {
```

```
                    bat "mvn -Dmaven.test.skip=true clean package"
```

```
                }
```

```
            }
```

```
        }
```

```
        stage('Docker Image Creation') {
```

```
            steps {
```

```
                dir('C:/Users/Swaroop/Desktop/EclipseWorkspace/Backend-aadhar') {
```

```
                    bat "docker build -t taxi --output type=docker ."
```

```
                }
```

```
            }
```

```
}

stage('Docker Hub Push') {

    steps {





















        bat "docker tag taxi swaroopav21/capstone"
        bat "docker push swaroopav21/capstone"





















    }

}

}
```



- ▼  Backend-aadhar [boot]
  - ▼  src/main/java
    - >  com.sec
    - >  com.sec.controller
    - ▼  com.sec.entity
      - >  AdminEntity.java
      - >  Usercredentials.java
      - >  UserEntity.java
    - >  com.sec.repo
    - >  com.sec.service
  - >  src/main/resources
  - >  src/test/java
  - >  JRE System Library [JavaSE-17]
  - >  Maven Dependencies
  - >  src
  - >  target
  -  HELP.md
  -  mvnw
  -  mvnw.cmd
  -  pom.xml

- ▼  Application-test-aadhar
  - ▼  src/main/java
    - ▼  com.test
      - >  Aadhartest.java
  - >  src/main/resources
  - >  src/test/java
  - >  src/test/resources
  - >  JRE System Library [JavaSE-17]
  - >  Maven Dependencies
  - >  target/generated-sources/annotations
  - >  target/generated-test-sources/test-annotations
  - >  src
  - >  target
  - >  test-output
  -  chromedriver.exe
  -  Dockerfile
  -  Jenkins.txt
  -  pom.xml
  -  selenium-server-4.10.0.jar
  -  testng.xml

